REMARKS

This reply is submitted in response to the Office Action dated April 20, 2007. The amendments above and the remarks that follow address the points raised in the Office Action and, thereby are believed to place this application in condition for allowance.

Claim Objections

Claims 1, 3-10, and 12-16 are objected to due to a number of informalities. Applicants respectfully disagree with the Examiner's assertion that processes cannot communicate with each other. The fact that the first and second processes correspond to communication tasks does not preclude their ability to communicate with one another. In fact, claim 1 recites that the first process sends to the second process, via the protocol translator, a set of executable instructions.

In general, processes typically need to communicate with one another, and they frequently do. For example, the output of a one process can be passed to another process. See, e.g., the enclosed page 100 of Modern Operating Systems, authored by Tanenbaum, Andrew S., Second Edition, Prentice Hall, 2001.

Thus, withdrawal of the claim objections is respectfully requested.

Claim Rejections under 35 U.S.C. §103

Claims 1 and 5

Claims 1 and 5 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,430,503 of McBurney in view of U.S. Patent No. 6,570,530 of Gaal et al.

Claim 1 is directed to a communications device for detecting user transmitted symbols encoded in spread spectrum waveforms ("user waveforms"). The device includes a first operating system supporting execution of a first process corresponding to a first set of communication tasks for detecting user transmitted symbols encoded in the user waveforms. The device further includes a second operating system supporting execution of a second process corresponding to a second set of communication tasks for detecting user transmitted symbols encoded in the user waveforms. The first and second operating systems differ. The device also includes a protocol translator coupled to the first and second processes and translating communications in between. The first process sends to the second process, via the protocol translator, a set of executable instructions for performing at least a portion of said second set of communication tasks. The second process generates a matrix as a result of executing the set of instructions.

McBurney is generally directed to a GPS navigation system having a measurement platform for receiving raw positioning signals, and a user platform for generating navigation solutions using the information from the measurement platform. A channel control sub-system is used to process the signals received by the measurement platform and transmit its results to the user platform.

The Examiner has failed to establish a prima facie case of obviousness because the combination of references does not teach the claimed invention. While claim 1 recites that the first process sends *executable instructions* to the second process, there is no indication in McBurney that the measurement platform sends executable instructions to the user platform. Rather, in McBurney, the channel control sub-system transmits *intermediate data* to the user platform (See Col. 4, lines 5-8 of McBurney). In other words, it simply moves data, not instructions, from the measurement platform to the user platform. Nor is there any indication that the channel control sub-system functions as a protocol translator, e.g., to translate executable instructions. Further, Gaal does not remedy this deficiency of McBurney as Gaal does not teach the use of multiple operating systems with a protocol translator to translate communication, including executable instructions, between them.

Thus, claim 1 and claim 5 which depends therefrom is patentable over the combination of McBurney and Gaal.

Claims 6-8

Claims 6-8 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,430,503 of McBurney in view of U.S. Patent No. 6,570,530 of Gaal et al. and in further view of U.S. Patent No. 5,590,284 of Crosetto.

Claims 6-8 depend from claim 1, and hence include all its features. As discussed above, claim 1 is patentable over McBurney and Gaal. Crosetto does not remedy the deficiencies of McBurney and Gaal. Crosetto is directed to a serial network consisting of a master processor and a number of slave processors controlled by the master. Crosetto, however, does not teach a protocol translator coupled to the first and second processes and translating communications in between. Thus, claims 6-8 are patentable over the combined references.

Claim 9

Claim 9 stands rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,430,503 of McBurney in view of U.S. Patent No. 6,570,530 of Gaal et al. in view of U.S. Patent No. 5,590,284 of Crosetto in further view of U.S. Patent No. 6,885,338 of Gaus, Jr. et al.

Claim 9 depends from claim 1, and hence include all its features. As discussed above, claim 1 is patentable over McBurney and Gaal. Crosetto does not remedy the deficiencies of McBurney and Gaal, as discussed above. Nor does Gaus, Jr. remedy the deficiencies of McBurney and Gaal. Gaus, Jr. is directed to a serial network consisting of a master processor and a number of slave processors controlled by the master. Gaus, Jr., however, does not teach a protocol translator coupled to the first and second processes and translating communications in between. Thus, claim 9 is patentable over the combined references.

Claims 3, 4, 10, and 12-14

Claims 3, 4, 10, and 12-14 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,430,503 of McBurney in view of U.S. Patent No. 6,570,530 of Gaal et al. and in further view of U.S. Patent No. 6,885,338 of Gaus, Jr. et al.

Claims 3 and 4 depend from claim 1, and hence include all its features. As discussed above, claim 1 is patentable over McBurney and Gaal. Gaus, Jr. does not remedy the deficiencies of McBurney and Gaal, as discussed above. Thus, claims 3 and 4 are patentable over the combined references.

The arguments above with respect to claim 1 apply with equal force to establish that independent claim 10 is also patentable. For example, similar to claim 1, claim 10 recites a protocol translator coupled to the first and second processes and translating communications in between, features not taught by McBurney or Gaal. Thus, claim 10 is patentable over the combination of McBurney and Gaal. Claims 12-14 depend from claim 10 and hence include all its features. Further, Gaus, Jr. does not remedy the deficiencies of McBurney and Gaal. Thus, claim 10, and claims 12-14 which depend therefrom, are patentable over the combined references.

Conclusion

In view of the above amendments and remarks, Applicants respectfully submit that the claimed invention is in condition for allowance. Applicants therefore kindly request reconsideration and allowance of the pending application.

Dated: July 6, 2007

Respectfully submitted,

By [illegible]

Reza Mollaaghababa
Registration No.: 43,810
NUTTER MCCLENNEN & FISH LLP
World Trade Center West
155 Seaport Boulevard
Boston, Massachusetts 02210-2604
(617) 439-2514
(617) 310-9514 (Fax)
Attorney for Applicant

1641441.1

more stack automatically. When a process has multiple threads, it must also have multiple stacks. If the kernel is not aware of all these stacks, it cannot grow them automatically upon stack fault. In fact, it may not even realize that a memory fault is related to stack growth.

These problems are certainly not insurmountable, but they do show that just introducing threads into an existing system without a fairly substantial system redesign is not going to work at all. The semantics of system calls may have to be redefined and libraries have to be rewritten, at the very least. And all of these things must be done in such a way as to remain backward compatible with existing programs for the limiting case of a process with only one thread. For additional information about threads, see (Hauser et al., 1993; and Marsh et al., 1991).

2.3 INTERPROCESS COMMUNICATION

Processes frequently need to communicate with other processes. For example, in a shell pipeline, the output of the first process must be passed to the second process, and so on down the line. Thus there is a need for communication between processes, preferably in a well-structured way not using interrupts. In the following sections we will look at some of the issues related to this **InterProcess Communication** or **IPC**.

Very briefly, there are three issues here. The first was alluded to above: how one process can pass information to another. The second has to do with making sure two or more processes do not get into each other's way when engaging in critical activities (suppose two processes each try to grab the last 1 MB of memory). The third concerns proper sequencing when dependencies are present: if process *A* produces data and process *B* prints them, *B* has to wait until *A* has produced some data before starting to print. We will examine all three of these issues starting in the next section.

It is also important to mention that two of these issues apply equally well to threads. The first one—passing information—is easy for threads since they share a common address space (threads in different address spaces that need to communicate fall under the heading of communicating processes). However, the other two—keeping out of each other's hair and proper sequencing—apply equally well to threads. The same problems exist and the same solutions apply. Below we will discuss the problem in the context of processes, but please keep in mind that the same problems and solutions also apply to threads.

2.3.1 Race Conditions

In some operating systems, processes that are working together may share some common storage that each one can read and write. The shared storage may be in main memory (possibly in a kernel data structure) or it may be a shared file;